一种基于海绵函数的快速伪随机序列生成方法 *

赵 磊[a, b]，郑 东[a, b]，任 方[a, b†]

(西安邮电大学 a. 通信与信息工程学院; b. 无线网络安全技术国家工程实验室, 西安 710121)

摘 要：海绵函数使用较短的密钥和初始向量，因而作为一种新型的伪随机生成器结构被使用。针对 Meziani 等人提出的 2SC（sponge code-based stream cipher）伪随机序列生成方法效率低、速度慢等问题，结合编码理论，提出了一种基于海绵函数的快速伪随机序列生成方法。使用一个通用的状态转换将其安全性归约为正则校验子译码问题，但是其计算能力相比于正则编码更好。理论分析和实验结果表明，该伪随机序列生成器保留了海绵函数的特性，但是效率大大提高。对于 160bits 的安全级别，其速度比原方案提高到了 5 倍以上。同时 NIST 统计测试及序列的平衡度、互相关性等测试结果表明生成的伪随机序列具有良好的随机特性。

关键词：海绵函数；伪随机序列；编码理论；正则字；校验子译码

中图分类号：TP309.2 doi: 10.19734/j.issn.1001-3695.2018.05.0451

Fast pseudo-random sequence generation method based on sponge function

Zhao Lei[a, b], Zheng Dong[a, b], Ren Fang[a, b†]

(*a. School of Telecommunication & Information Engineering*, *b. National Engineering Laboratory for Wireless Security*, *Xi'an University of Posts & Telecommunications*, *Xi'an* 710121, *China*)

Abstract: The sponge function uses shorter keys and initial vectors, so it is used as a new pseudo-random generator structure. This paper proposed a fast pseudo-random sequence generation method based on spongy function in combination with the coding theory, to solve the problems of low efficiency and slow speed of 2SC (sponge code-based stream cipher) pseudo-random sequence generation method proposed by Meziani et al. uses a generic state transformation which is reducible to the Regular Syndrome Decoding problem(RSD) , but has better computational characteristics than the regular encoding. Theoretical analysis and experimental results show that the pseudo-random sequence generator preserves the characteristics of sponge function, but it improved the efficiency greatly. For the security level of 160bits, its 5 times faster than the original scheme. At the same time, NIST statistical test, sequence balance and cross correlation test results show that the generated pseudo-random sequence has good random characteristics.

Key words: sponge function; pseudo-random sequence; coding theory; regular word; syndrome ddecoding

0 引言

密码技术是网络安全的根基。在网络行为已经渗透到社会生活每个领域的今天，无论是网上银行、电子商务还是电子邮件、即时消息服务，密码技术无时无刻不在保护着用户的信息安全。流密码[1]作为一种主流的加密方式，加解密实现方式简单、速度快、理论相对成熟，因而在许多实际中应用，特别是无线通信标准，如 IEEE 802.11 b 和蓝牙。伪随机序列常作为流密码的密钥流，与待加密的明文流进行简单的“异或”运算得到密文流。这是一种“一次一密”的加密方式，能够得到最高的安全性－完善保密性。因而密钥流的生成成为了流密码算法的关键，当前，主要的密钥流生成器都是基于伪随机数生成器设计和实现的，因此设计性能良好的伪随机数发生器（pseudo-random number Generator，PRNG）已经成为密码学领域的一个研究热点[2-3]。

作为在流密码中的应用，伪随机序列不仅需要具备良好的统计特性和有效性，同时需要可证明安全的伪随机序列生成器，即需要把伪随机序列的安全性规约到特定的困难问题上。在已有的伪随机序列生成方法中，基于线性反馈移位寄存器（linear feedback shift register,LFSR）的生成方法是常见的方法之一，然而存在许多安全弱点，线性复杂度低。为了抵抗攻击，在 20 世纪 80 年代初，Blum[4]等人提出了基于大整数分解问题的可证明安全的伪随机序列生成器，随后，Kaliski[5]提出了另一种方案，其安全性依赖于离散对数问题的难解性。Alexi 等人[6]提出了基

收稿日期：2018-05-18；**修回日期：**2018-07-11 **基金项目：**国家自然科学基金资助项目（61472472）；西安邮电大学研究生创新基金资助项目（CXJJ2017014）

作者简介：赵磊（1993-），男，陕西延安人，硕士研究生，主要研究方向为信息安全（1045571121@qq.com）；郑东（1964-），男，教授，博士，主要研究方向为密码学、云存储安全；任方（1981-），男（通信作者），副教授，博士，主要研究方向为密码学与网络安全（renfang_81@163.com）.

于 RSA 问题的可证明安全的伪随机序列生成方法。

尽管上面所提到的伪随机序列生成方法是可证明安全的，但它们在量子攻击[7]下都将失去原有的安全性。因此，设计基于其他假设的伪随机数生成器显得尤为重要，这也是近年来抗量子密码技术研究的方向之一。最早提出抗量子攻击的伪随机序列生成器是 Impagliazzo 等人[8]，提出了基于子集和问题的可证明安全的伪随机序列生成方法。之后，Fisher 和 Stern[9]提出了一种基于校验子译码(SD)问题的伪随机序列生成器，但是存在两个缺陷：运算速度慢，用于计算校验子的矩阵占用内存大，因此难以得到应用。Gaborit[10]等人在 2007 年提出了新的方案，称为 SYND,是对文献[9]的一种改进，并将其安全性规约到 SD 问题。2011 年，Meziani[11]等人利用海绵结构提出了一种新的基于校验子译码问题的伪随机序列生成方法，称为 2SC。这种密码结构相比于 SYND，在性能指标上更加有效，而且只需小的密钥大小和初始向量就可以获得较高的安全级别。但是其最大的缺点是需要大的矩阵。2016 年，Gaborit[12]等人基于 Rank Metric 码的校验子译码问题的困难性构造了一种伪随机序列生成器。

本文重新考察了由 Meziani 等人提出的 2SC 方案，提出了一种对 2SC 改进的高效算法。保留了海绵结构，使得密钥和初始向量较短。相比于 2sc，本文摒弃了将码字编码为正则字[13]的过程，而使用哈希算法中压缩函数的构造思想，将其转换为一种可以规约为正则字校验子译码问题的编码方式。从而使得伪随机序列生成速度更快。

1　编码理论

本节主要回顾了基于编码的密码学基础，简单描述了编码理论中的常见困难问题，最后介绍了基于编码的哈希函数。

1.1　线性分组码

有限域 F_q 上的一个 (n,k) 线性分组码 C 定义为 n 维线性空间 F_q^n 的一个 k 维线性子空间。其中 F_q^n 中的向量称为字，而 C 中的向量称为码字(codeword)，n 称为分组码的码长，k 称为分组码的维数。

定义 1　一个 (n,k) 线性分组码 C 的生成矩阵是一个 $k\times n$ 的矩阵 G，其中 G 的行向量构成了 C 的一组基。

线性分组码 C 的生成矩阵 G 不唯一，但是不同的生成矩阵可以通过初等行变换相互转换，即若 G 是 C 的一个生成矩阵，P 是一个初等矩阵，则 PG 也是 C 的一个生成矩阵。

定义 2　一个 (n,k) 线性分组码 C 的校验矩阵是一个 $(n-k)\times n$ 的矩阵 H，其中 H 的任意行向量与 C 的生成矩阵 G 的任意行向量正交，即 $HG^{\mathrm{T}}=0$。

线性分组码的校验矩阵不唯一且可以通过初等行变换相互转换。长为 n 的向量 c 是 C 的一个码字的充分必要条件是 $Hc^{\mathrm{T}}=0$。对于任意的字 c，将 Hc^{T} 称为 c 的校验子。

定义 3　线性分组码的两个码字，$u=(u_1,u_2,\cdots,u_n)$，$v=(v_1,v_2,\cdots,v_n)$ 的 Hamming 距离 $d(u,v)$ 定义为 u 和 v 的不同分量的个数，即 $d(u,v)=|\{i:u_i\neq v_i\}|$。

码字 u 的 Hamming 重量 $w(u)$ 定义为 u 的非零分量的个数，码 C 的非零码字的最小 Hamming 重量称为 C 的最小距离，记作 $d_{\min}$。码的最小距离决定了码的纠错能力，一般来说，线性分组码的纠错能力 t 满足。

定义 4　给定长度为 n 重量为 w 的字 x，将其转换成 w 个长度为 n/w 的块，每一块都有固定的“1”，称这样的码字为正则字。

1.2　困难问题

在基于编码的密码学中，其安全性大多都规约到以下困难问题中。

Q1　校验子译码问题(Syndrome Ddecoding,SD)

给定有限域 F_q 上的一个 $(n-k)\times n$ 的矩阵 H，向量 $s\in F_q^{n-k}$，整数 $w>0$，问是否存在一个字 $x\in F_q^n$，其重量小于等于 w，使 $Hx^{\mathrm{T}}=s$。

Q2 寻找码字问题(Codeword Finding,CF）

给定有限域 F_q 上的一个 $(n-k)\times n$ 的矩阵 H,整数 $w>0$，问是否存在一个非零的字 $x\in F_q^n$，其重量小于等于 w,使 $Hx^{\mathrm{T}}=0$。

SD 问题和 CF 问题在文献[14]中已被证明是 NP-完全问题。SD 问题的一个特例就是 RSD 问题，该问题在文献[13]中证明为 NP-完全问题，具体描述如下。

Q3 正则校验子译码问题(Regular Syndrome Decoding,RSD)

给定有限域 F_q 上的 w 个 $(n-k)\times n/w$ 矩阵 H_i，向量 $s\in F_q^{n-k}$，整数 $w>0$，问是否存在 w 个向量 h_i, h_i 是 H_i 的列向量，使 s 等于这些向量之和。

1.3　基于编码的 Hash 函数构造

Hash 函数的构造中，最核心的部分是压缩函数 g。基于编码困难问题的压缩函数 g 的构造方法如下。

随机选择一个 (n,k) 码字 x，其中：

$$n=2^m,k=n-mt \tag{1}$$

选择正整数 $w|n$，显然有

$$w=2^{m'}(m'<m) \tag{2}$$

另取

$$l=\frac{n}{w}=2^{m-m'} \tag{3}$$

则可以将长度为 n 的每一个字分成等长的 w 个块，每一块包含 l 个比特。

随机选取一个 $(n-k)\times n$ 的校验矩阵 H,可以将 H 分成 w 个子矩阵，即

$$H=(H_1,H_2,\cdots H_w) \tag{4}$$

其中每一个子矩阵

$$H_i = (h_j)(i=1,2,\cdots w; j=(i-1)l+1,(i-1)l+2,\cdots il) \quad (5)$$

h_j 为矩阵 H 的第 j 列。

定义压缩函数

$$g: \ F_2^e \to F_2^r,$$

$$e = w\log_2^l, r = n-k = mt \quad (6)$$

对于任意的 $x \in F_2^e$，将 x 也按照上述的划分方法分成 w 个块，即 $x=(x_1,x_2,\cdots,x_w)$，其中 $x_i \in F_2^{\log_2^l}$，将 x_i 转换成 0 至 $l-1$ 之间的数，选择矩阵 H_i 的第 x_i+1 列，即 $h(i-1)l+x_i+1$，计算 $z=\sum_{i=1}^{w} h(i-1)l+x_i+1$，则函数的输出：

$$g(x) = z \quad (7)$$

定理 1 上述的压缩函数 g 的输出等价于计算一个长为 n，重量为 w 的正则字的校验子，即对于任意的 $g(x)$，存在正则字 c，使得 $Hc^{\mathrm{T}} = g(x)$。证明参考文献[15]。

2　基于 Sponge 函数的伪随机序列生成方法-2SC

2.1　Sponge 函数（海绵函数）

Sponge 函数最先由 Peeters 等人在文献[16]提出，它为迭代哈希函数设计提供了一种新的方法。它使用有限的状态，接收任何长度的输入位元流，然后可以满足任何长度的输出。

Sponge 函数由三个部分组成，如图 1 所示。

a）一个内存状态 S，包含 s 个位元。内存状态会分成两个区块，R（大小为 r 位元）与 C（大小为 $c=s-r$ 位元）。这里的参数 r 又叫做转换率（bitrate），而 C 叫做容量（capacity）。

b）一个能置换或者转换内存状态，固定大小的转换函式 f。

c）一个填充函数（padding function）P。填充函式会在输入里面增加足够的长度，让输入的位元流长度变成 r 的整数倍。因此填充过后的输入可以被切成长度为 r 的整数个分段。

Sponge 函数的具体运行过程如下：

a） S 先初始化为零。

b）输入经过填充函式处理。

c）填充后输入的前 r 个位与 R 进行异或运算。

d） S 经过函式 f 转换成 $f(S)$。

e）如果填充后输入还有剩余，下一 r 位则与上一步 R 进行异或运算。

f） S 转换成 $f(S)$。

…

这个过程一直重复到所有的输入都用完为止，称为 absorbing 阶段。(在海棉的譬喻里面，表示被函数“吸收”了)。

输出块 Z_i 的数量由用户选择，这一阶段称为 squezzing 阶段（“挤出”），具体过程如下：

a）通过 absorbing 阶段得到新的内存状态，此时 R 表示输出的前 r 个位元。

b）如果需要更多输出，则把 S 转换成 $f(S)$。

c）此时 R 则表示输出的下 r 个位元。

…

这一过程会重复到满足输出所需要的长度为止。需要注意的是，输入绝对不会与 C 这部分的内存作异或运算，而且这一部分内存也不会直接被输出。这一部分的内存仅仅只和转换函式相关。在杂凑里面，防止撞击攻击(Collision attack)或者原像攻击(preimage attack)是依靠这段内存做到的。

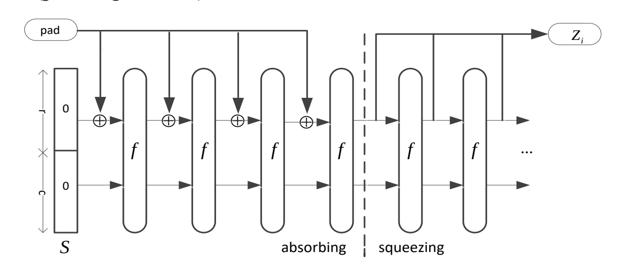


图 1　Sponge 函数结构

Fig.1　Sponge function structure

2.2　2sc 方案结构

基于上节所提到的 Spange 函数，Meziani 等人设计了一种伪随机序列生产方法。令 K 和 IV 分别表示密钥和初始向量，并且 $r=|K|=|IV|$，内部状态 S 的大小 $s=w\log 2(n/w)$，其中 $n>w, n/w=2^l$。

Initialization 初始化函数 f 通过输入密钥 K 和初始向量 IV，计算并且返回初始状态，定义如下：

$$f: F_2^{|k|} \times F_2^{|IV|} \to F_2^s \quad (8)$$

$$(x_1,x_2) \to f(x_1,x_2) = f_1\left(\left(f_1^{[r]}(x_1 \mid 0^c)\oplus x_2, f_1^{[c]}(x_1 \mid 0^c\right)\right)$$

其中: “|” 表示级联，“ 0^c ” 是大小为 c 的全零向量。 $f_1^{[r]}$ 表示 $f_1(\cdot)$ 的前 r 位比特，$f_1^{[c]}$ 表示 $f_1(\cdot)$ 的剩余 c 位比特。校验子映射 f_1 定义如下：

$$x \to f_1(x) = \phi(x)\cdot H_1^T \quad (9)$$

这里，函数 $x \to \phi(x)$ 指正则编码，将 s 位的字符串转换成长度为 n 和重量为 w 的正则字。矩阵 H_1 是大小为 $s\times n$ 的二元随机矩阵，该初始化步骤需要计算两个函数并进行两个异或运算。因此，当每个安全级别选择最优参数 (n,s,w)，那么比 SYND 方案更加有效。通过初始化过程得到初始状态 e_0。

Update 在此步骤中，将使用一个新的随机函数 g 来更新内部状态(比如 N 次)。g 运行的次数是由用户选择的，更新的次数也因此影响着结构的安全性和效率。函数 g 的定义如下:

$$g: F_2^s \to F_2^s$$

$$x \to g(x) = \phi(x) \cdot H_2^T \tag{10}$$

其中: H_2 同样是一个 $s \times n$ 的二元随机矩阵，函数 $x \to \phi(x)$ 指正则编码，与初始化步骤中的过程相同。

Squeezing 通过更新过程得到 e_i，2SC 最终产生每组大小为 r 比特密钥流 z_i ($i \geq 1$)，描述如下：

a) $z_1 = g(e_0)^{[r]}$,表示初始状态 e_0 经过更新函数得到新的内部状态 e_1， e_1 的前 r 位输出即为 z_1。

b)对于 $i \geq 2$， $e_i = g(e_{i-1})$,其中 e_i 是更新状态下 g 的 i 次迭代的输出。

上述所描述的结构如图 2 所示。

3　一种基于海绵函数的快速伪随机序列生成方法

在实验仿真过程中发现上节所介绍的伪随机序列生成方法在正则编码 Niederreiter 变换[17]，即 $x \to \phi(x)$ 过程中需要消耗大量时间，因此希望通过另一种编码方式将其性能提高，同时不降低安全性。基于编码的哈希函数的构造方法思想将上述方案进行改进，具体描述如下。

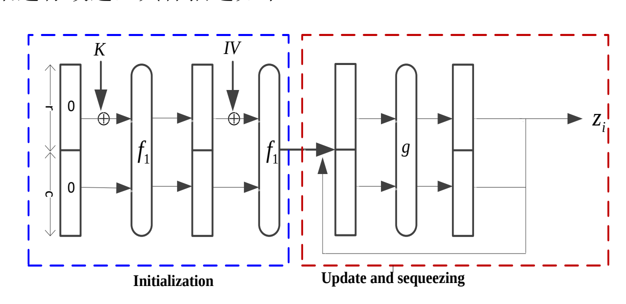


图 2　2SC 密钥流生成器结构图

Fig.2　2SC key stream generator

给定输入 x，由 w 块 $x_1, x_2, \cdots, x_w$ 组成， x_i （$1 \leq i \leq w$）的大小为 b 比特，本文首先通过函数 f 来给每个块提供数据，得到输出 y_i，将 $y_1, y_2, \cdots, y_w$ 进行异或运算最终得到输出。结构如图 3 所示。

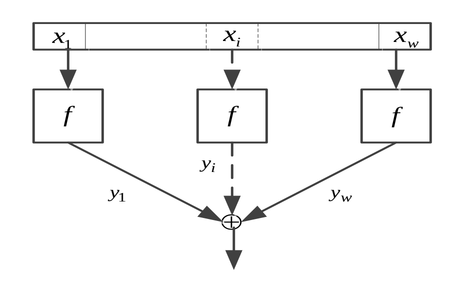


图 3　$f(x)$ 函数结构

Fig.3　$f(x)$ function structure

f 函数具体描述为：选取大小为 $wb \times 2^b$ 的二元随机矩阵 H_1，将其分成 w 个大小为 $wb \times 2^b$ 子矩阵 $H_1, H_2, \cdots, H_w$ （$H = H_1 | H_2 | \cdots | H_w$），子矩阵 $H_i = \left(h_i^{(0)}, h_i^{(1)}, \cdots, h_i^{(2^b-1)}\right)$，其中，$h_i^{(j)} \in F_2^{wb}$， $j \in \{0, 1, \cdots, 2^b - 1\}$。令 x_i 的十进制等于 j，那么 x_i 通过函数 f 得到 $y_i = h_i^{(j)}$，每一个 y_i 则有 2^b 种可能，按照上述方法重新定义了函数 f_1 和函数 g：

$$f_1(x) = a_1^{(x_1)} \oplus a_2^{(x_2)} \oplus \cdots \oplus a_w^{(x_w)} \tag{11}$$

$$g(x) = b_1^{(x_1)} \oplus b_2^{(x_2)} \oplus \cdots \oplus b_w^{(x_w)} \tag{12}$$

其中： A 和 B 是大小为 $wb \times w \cdot 2^b$ 的二元随机矩阵， A_i 和 B_i 分别是 A 和 B 的子矩阵， $a_i^{(j)}$ 则表示矩阵 A 的第 i 个子矩阵的第 $j+1$ 列， $b_i^{(j)}$ 则表示矩阵 B 的第 i 个子矩阵的第 $j+1$ 列。

通过以下的一个例子将上述的方法展示。首先，令 $w = 3$， $b = 2$。矩阵 A 则是 $(3 \cdot 2) \times (3 \cdot 2^2) = 6 \times 12$ 阶矩阵。随机选取一个符合上述要求的矩阵 A 如下：

$$A = \left[\begin{array}{cccc|cccc|cccc} a_1^{(0)} & a_1^{(1)} & a_1^{(2)} & a_1^{(3)} & a_2^{(0)} & a_2^{(1)} & a_2^{(2)} & a_2^{(3)} & a_3^{(0)} & a_3^{(1)} & a_3^{(2)} & a_3^{(3)} \\ \hline 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

假设输入 $x = [10 | 01 | 00]$，将 x 分成 w 组并表示为十进制，则 $x' = [210]$。根据公式 $z_i = (x_i + 1) + (i - 1)2^b$，计算得到 $z_1 = 3$， $z_2 = 6$， $z_3 = 9$。在二进制记数法中， z_i 表示 z 的非零项的位置，即 $z = [0010 | 0100 | 1000]$。因此可以验证：

$$g(x) = a_1^{(2)} \oplus a_2^{(1)} \oplus a_3^{(0)} = [001111]^T = A \cdot z^T \tag{13}$$

4　安全性分析

本章将从两个角度来分析该算法的安全性。在理论上，很难逆向恢复出密钥 K 和初始向量 IV；而在实际攻击中，需要敌手解决 SD 问题，目前已知的有两种可以有效的针对 SD 问题的攻击：信息集译码攻击（ISD）和广义的生日攻击（GBA）。

4.1　理论安全性

由第 3 节所描述的转换函数 $f_1(x)$ 和 $g(x)$，定义一个通用的函数 D：

$$D(x) = a_1^{(x_1)} \oplus a_2^{(x_2)} \oplus \cdots \oplus a_w^{(x_w)}$$

$$\forall x = (x_1, x_2, \cdots, x_w) \in F_2^{wb} \tag{14}$$

在这个转换过程中，A 是大小为 $wb \times w \cdot 2^b$ 的二元随机矩阵，A_i 是 A 的子矩阵，$a_i^{(j)}$ 则表示矩阵 A 的第 i 个子矩阵的第 $j+1$ 列（$j = 0,1,\cdots 2^b - 1$）。需要证明两点：

a）对于每一个输入码字 x，存在一个正则字 z，使得 $D(x) = A \cdot z^T$。

b）由 $y = D(x)$ 求解 x 等价于寻找一个正则字 z，重量小于或等于 w，使得 $A \cdot z^T = y$。这正是 RSD(n,r,w) 对 $r = wb$ 和 $n = w2^b$ 的实例化。因此，改进后方案的安全性可以规约到 RSD 问题。

证明　a）首先 $A = A_1 | A_2 | \cdots | A_w$，每一个子矩阵 A_i 的大小为恰为 $wb \times 2^b$，每个子矩阵的列 j 从 $a_i^{(0)}, a_i^{(1)}, \cdots a_i^{(2^b-1)}$。

对于正则字 z，长度 $n = w2^b$，重量为 w，令 $z_1, z_2, \cdots z_w$ 分别表示其非零项的位置（因为这个字是正则字，所以每个 z_i 在 $(i-1)2^b + 1$ 和 $i2^b$ 之间是唯一的值）。

令输入 $x = (x_1, x_2, \cdots, x_w)$，则 x_i 与 z_i 存在这样的关系：$z_i = (x_i + 1) + (i-1)2^b$，那么由 z 到 x 的逆向变换如下所示：

$$\begin{cases} x_1 \equiv z_1 - 1 \pmod{2^b} \\ x_2 \equiv z_2 - 1 \pmod{2^b} \\ \cdots \\ x_w \equiv z_w - 1 \pmod{2^b} \end{cases} \tag{15}$$

因此，很容易验证：

$$\begin{aligned} A \cdot z^T &= a^{(z_1-1)} \oplus a^{(z_2-1)} \oplus \cdots \oplus a^{(z_w-1)} \\ &= a_1^{(z_1-1)(\mathrm{mod}\ 2^b)} \oplus a_2^{(z_2-1)(\mathrm{mod}\ 2^b)} \oplus \cdots \oplus a_w^{(z_w-1)(\mathrm{mod}\ 2^b)} \\ &= a_1^{(x_1)} \oplus a_2^{(x2)} \oplus \cdots \oplus a_w^{(x_w)} \\ &= \mathrm{D}(x) \end{aligned}$$

b）已经证明了对于每个输入 x，可以找到一个重量为 w 的正则字 z，使得 $A \cdot z^T = D(x)$。假设存在一个可以逆向求出 $D(x)$ 的敌手，即给定 $y = D(x)$，敌手输出 x。那么给定一个矩阵 A 和一个值 $y = D(x) = A \cdot z^T$，同样的敌手便可以输出正则字 z。这正是 RSD(n,r,w) 的一个实例，对于 $r = wb$ 和 $n = w2^b$。因此，可以把该算法的安全性规约到 RSD 问题的困难性上，而 RSD 问题同 SD 问题一样为 NP 完全问题。

4.2　实际安全性

在实践中，敌手将面临两个问题。一方面，敌手获得 r 位的输出，但无法得到剩余的 $c = b - r$ 位，容量越大，系统越安全。另一方面，即使成功地猜测了这些比特，对手也必须解决 RSD 问题的一个实例。对于一个选择适当的参数集，有效解决 RSD 问题和 SD 问题一样困难。事实上，所有已知的 SD 攻击都是完全指数型的，只有两种算法可以攻击基于 SD 问题的系统：信息集译码（ISD）[10]和广义生日算法（GBA）[18]。在文献[19]中提出了针对基于编码的密码系统的最新的 GBA 算法并将用于选择 2SC 的安全参数。

注意，还有一种攻击称为 Memory trade-off attacks，这种攻击最初是在[20]中引入的，作为攻击分组密码的通用方法。为了避免这种攻击，初始向量 IV 应该至少和密钥 K 一样大，而 s 至少应该是密钥的两倍。

根据这一节及前两节的描述，本文从结构上对比 2SC 方案和本文的方案，如表 1 所示。在下一章将对其进行具体的实验仿真对比。

表 1　2SC 方案和本文方案的结构对比

Table 1 Structural comparison between 2SC scheme and proposed scheme

方案	循环结构	编码算法	依赖的困难问题	问题性质	特点
2SC	Sponge 结构	Niederreiter 变换	SD 问题	NP 完全	耗时较长
本文	Sponge 结构	基于编码的 hash 变换	RSD 问题	NP 完全	耗时较短

5　实验仿真

对改进后的伪随机序列生成算法进行分析，并与原始的 2SC 算法进行比较，同时测试生成序列的随机性。

5.1　算法的效率分析

基于 4 章所提出的方案，利用 Python 语言对其过程进行仿真实现。(n,r,w) 参数选取必须在已知的攻击下具有高效率和高安全性。首先，根据 Time Memory Trade-Off attacks，S 至少应该是密钥的两倍。

$$s = r + c = w\log_2(n/w) \geq 2|IV|$$

$$|K| = |IV| = r \tag{16}$$

选取最优值 $b = \log_2(n/w) = 8$，对于每个安全级别 λ，解决 RSD 问题的复杂度至少为 2^λ。实验测试了不同安全级别下，产生 $r \times N$（N 表示循环次数）比特伪随机序列所需要的时间。表 1 给出了在几个安全级别上，通过选取最优的参数集 (n,r,w) 而获得序列的测试结果。需要注意的是，在仿真实现过程中，只使用了随机的二元矩阵，而未使用任何特定的结构。但是当奇偶校验矩阵为准循环矩阵时，可以找到提供相同安全级别的参数[10]。

表 2 中的结果在 Pycharm 中利用 Python 所实现，操作系统为 windows10，主频 2.3GHz。对于不同的安全级别，通过循环了 10 次产生 $r \times 10$ 位的数据。可以看到，随着校验矩阵 n 的增大，生成序列的时间相应的增加，但是安全性也随之提高。表 3 将 2SC 算法实现结果与新的算法实现结果进行对比分析，取 $N = 100$。

由表 2 可以得到以下结论。对于 80 bit 的安全级别，生成 9.6×10^3 bit 的伪随机序列，本文的方法比与原先的 2SC 方法速度提高了 3.19 倍；对于 120 bit 的安全级别，生成 1.44×10^4 bit 的伪随机序列，本文的方法比与原先的 2SC 方法速度提高了 3.02 倍；对于 160bits 的安全级别，生成 1.92×10^4 bit 的伪随机

序列，本文的方法比与原先的 2SC 方法速度提高了 5.75 倍；对于 400 bit 的安全级别，生成 3.04×10^4 bit 的伪随机序列，本文的方法比与原先的 2SC 方法速度提高了 4.5 倍。

表 2　不同安全级别下的测试结果

Table 2 Test results under different security levels

安全级别	n	s	w	c	|Key|/bits	时间/s
80	8192	256	32	160	96	0.213
120	12288	384	48	240	144	0.485
160	16384	512	64	320	192	0.835
200	20480	640	80	400	240	1.332
240	24576	768	96	480	288	1.868
280	28672	896	112	560	336	2.569

注:测试时间并未计算初始化时间

表 3　不同安全级别下性能对比

Table 3 Performance comparison under different security levels

安全级别	n	s	w	c	|Key|/bits	2SC/s	Our/s	比率
80	8192	256	32	160	96	7.6022	2.382	3.19
120	12288	384	48	240	144	14.756	4.886	3.02
160	16384	512	64	320	192	44.663	7.765	5.75
400	32768	1024	128	720	304	137.682	30.596	4.50

表 4　随机性测试结果

Table 4 Test results of Randomness

测试项	1 组测试值	2 组测试值	3 组测试值	4 组测试值	5 组测试值	6 组测试值
Frequency	0.644081	0.096914	0.153868	0.849309	0.379944	0.921932
Block Frequency	0.328677	0.800956	0.106402	0.668934	0.250088	0.867440
Cumulative Sums	0.782085	0.106226	0.207897	0.812076	0.572709	0.667091
Runs	0.107008	0.821789	0.782522	0.057691	0.206671	0.275709
Longest Run of Ones	0.270825	0.739232	0.348457	0.341358	0.566602	0.116106
Rank	0.675960	0.041286	0.284170	0.251623	0.275808	0.984463
Discrete Fourier Transform	0.435383	0.502925	0.183317	0.129989	0.876031	0.212023
Non-periodic Template	0.376496	0.237213	0.022931	0.524622	0.510644	0.630072
Overlapping Template	0.663812	0.234041	0.163913	0.962218	0.800859	0.441365
Universal Statistical	0.583128	0.324725	0.147473	0.491854	0.835538	0.850540
Approximate Entropy	0.609971	0.290362	0.237763	0.762622	0.564112	0.006234
Random Excursions	0.226235	0.343650	0.095483	0.502313	0.790699	0.425471
Random Excursions Variant	0.636178	0.049265	0.496225	0.385500	0.061161	0.566982
Serial	0.483546	0.272573	0.524176	0.277855	0.873444	0.694912
Linear Complexity	0.206177	0.978800	0.885160	0.820437	0.862190	0.361264

注：有下划线的测试值表示未通过测试

5.2　NIST SP800-22 测试分析

对算法产生的伪随机序列进行随机性测试，测试软件为美国国家标准与技术研究所的 NIST SP800-22 测试包 sts-2.1.1 测试软件[21]。软件的测试项目包含 15 项，序列的测试长度需要大于 10^6 bit，当测试值 $P_value\ge 0.01$ 时，认为序列在该项测试中为随机的；当测试值 $P_value<0.01$ 时，认为序列在该项的测试中为非随机的。现用 sts-2.1.1 测试软件对算法产生的 6.72×10^6 bis 的伪随机序列进行测试，测试分为 6 组，每组的长度为 10^6 bits。测试结果如表 4 所示。

由测试结果可以看出，15 个测试项的 P-value 值只有一个测试值未通过测试，其余都大于显著性水平 α（$\alpha=0.01$）。其中，Frequency Test 测试均值大于 0.5，表示伪随机序列中 0 和 1 的个数接近相等，序列的均衡性较好；Runs 测试均值为 0.375，说明序列中游程数接近真随机序列中游程的个数；Serial 测试均值大于 0.5，说明指定长度的子序列出现的次数趋于等概；Cumulative Sums 测试值 3 个都在 0.6 以上，说明 0 、1 的分布较为均匀；Linear Complexity 测试值 4 个都在 0.8 以上，说明序列的复杂度较高。因此，本文生成的伪随机序列具有良好的随机特性。

5.3　相关性分析

序列的自相关系数与互相关系数是评价序列随机性的常用统计量[22]。设伪随机序列 b_n，长度为 N ,则所得序列的自相关系数定义如下。

$$ac(m)=\frac{1}{N}\cdot\sum_{i=1}^{N-m}b_i b_{i+m} \tag{17}$$

其中，m 为步长，序列的自相关系数与步长有关，当步长 m 变化时，自相关系数变化越小，说明序列的随机性能越好。

当步长 m 为 0 时的理论值为 0.5，其余步长时，理论值为 0.25　设两个不同的伪随机序列分别为 b_{1n} 和 b_{2n}，$n=1,2\cdots,N$，序列 b_{1n} 和 b_{2n} 的互相关系数定义为:

$$cc(m)=\begin{cases}\frac{1}{N}\cdot\sum_{i=1}^{N-m}b_{1i}\cdot b_{2(i+m)} & 0\le m\le N\\ \frac{1}{N}\cdot\sum_{i=1}^{N-m}b_{1(i+m)}\cdot b_{2i} & -N\le m\le 0\end{cases} \tag{18}$$

互相关系数理论值为 0。序列的自相关和互相关系数值越接近理论值，说明序列的随机性越好。实验随机选取了两组长度为 $N=5000$ 的伪随机序列 c_1 和 c_2 进行测试，实验结果如图 4 所示。

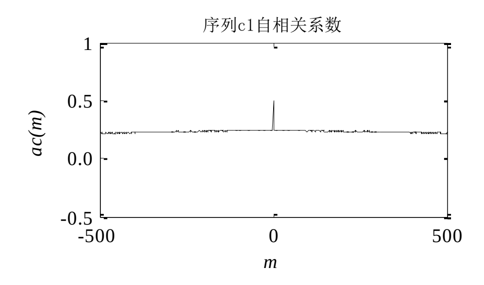
实验结果显示，序列的自相关性及互相关性分布均接近于理想状态，说明本文方法所得序列之间相关程度较小，具有较好的随机性。

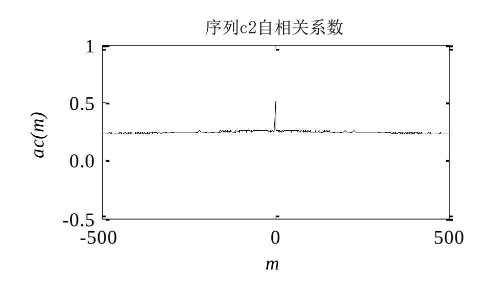
5.4　平衡性分析

设伪随机序列的长度为 N ,序列中 0 和 1 的个数分别为 P 和 Q，则序列的平衡度定义为

$$E(N)=\frac{|P-Q|}{N} \tag{19}$$

平衡度的值 $E(N)$ 越小，说明该序列中的 0 和 1 的个数越相近，随机性越好。实验选取了三组长度为 $N=38400$ 的生成序列 c_1、c_2 和 c_3 进行测试，图 5 给出了实验所得序列的平衡度分析。





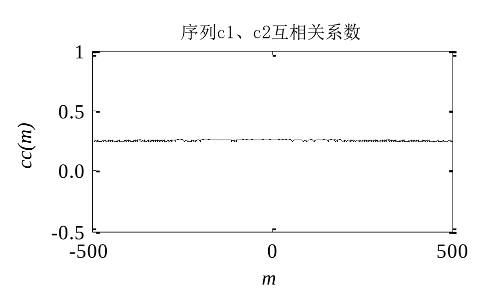


图 4　序列 c_1 和序列 c_2 的自相关及互相关系数

Fig.4　Autocorrelation and cross-correlation coefficients of sequences c_1 and sequences

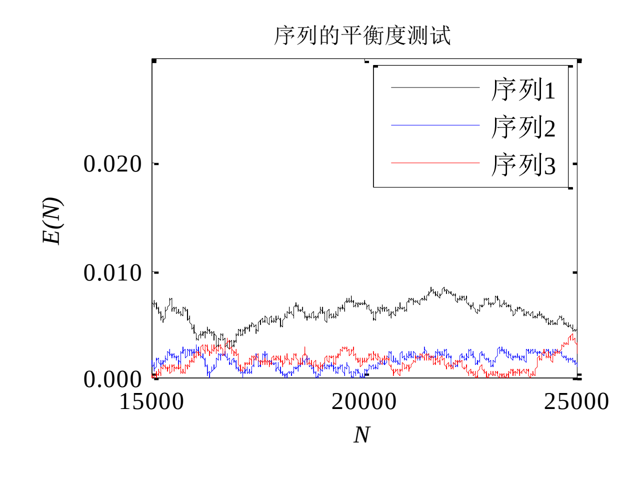


图 5　序列的平衡度测试

Fig.5　Sequence balance test

实验结果表明，三组伪随机序列的平衡度都在 0.010 以下，接近于 0。因此表明序列中的 0 和 1 的个数十分接近，因此序列的性能稳定，平衡度良好。

5.5　游程检验

游程即伪随机序列的一个子序列，由连续的 0 或 1 元素组成，其前一部分和后一部分的元素均和该元素不同，其中 0 或 1 连续出现的个数称为游程长度；随机序列中，任意长度的序列的 0 和 1 的个数近似相同，其中游程长度为 L 的序列约占总序列长度的 $1/2^L$，实验生成的伪随机序列越接近理想状态，则说明序列的随机性越好。实验所测序列的长度均为 $N = 38400$，对三组序列进行游程统计测试，具体统计情况如表 5~7 所示。

表 5　序列 c_1 的游程统计

Table 5 Travel statistics of sequence c_1

游程	游程长度							
	1	2	3	4	5	6	7	8
0	4788	2422	1209	568	312	153	78	48
1	4842	2356	1212	593	328	16	68	36
0/1	0.9888	1.0280	0.9975	0.9578	0.9512	1.1250	1.1471	1.3333
实测值	0.5012	0.2487	0.1260	0.0604	0.0333	0.0150	0.0076	0.0044
理论值	0.5000	0.2500	0.1250	0.0625	0.0313	0.0156	0.0078	0.0039

表 6　序列 c_2 的游程统计

Table 6 Travel statistics of sequence c_2

游程	游程长度							
	1	2	3	4	5	6	7	8
0	4855	2396	1168	607	323	155	74	46
1	4852	2485	1185	552	289	144	78	40
0/1	1.0006	0.9641	0.9856	1.0996	1.1176	1.0763	0.9487	1.1500
实测值	0.5027	0.2527	0.1218	0.0600	0.0316	0.0154	0.0078	0.0044
理论值	0.5000	0.2500	0.1250	0.0625	0.0313	0.0156	0.0078	0.0039

表 7　序列 c_3 的游程统计

Table 7 Travel statistics of sequence c_3

游程	游程长度							
	1	2	3	4	5	6	7	8
0	4828	2358	1193	590	330	156	84	43
1	4816	2403	1176	618	309	147	65	40
0/1	1.0025	0.9813	1.0145	0.9547	1.0680	1.0612	1.2923	1.0750
实测值	0.5020	0.2478	0.1233	0.0629	0.0333	0.0158	0.0078	0.0043
理论值	0.5000	0.2500	0.1250	0.0625	0.0313	0.0156	0.0078	0.0039

根据三组实验结果可知，生成的伪随机序列的游程统计接近理论值，其中 0 和 1 的个数接近相等。不同游程长度的序列占总序列的百分比与理论值相当。因此本文实验所得的序列符合游程统计特性。

6　结束语

本文提出了一种基于海绵函数的快速伪随机序列生成方法，是对 2sc 伪随机序列生成方法的改进。该方法一方面使得密钥和初始向量长度缩短，另一方面将码字编码为正则字的过程转换为一种可以规约为正则字校验子译码问题的编码方式，其计算速度相比于原来大大提高。实验仿真结果表明本文的方法在相同安全级别下将比原来序列生成速度至少提高 3 倍。NIST 的测试结果表明本文方法生成的伪随机序列具有良好的随机性，序列的自相关性、互相关性、平衡度、游程统计都符合理论值。同时，方案的安全性依赖于 RSD 问题，这使得该方法可以抵抗现有的量子攻击。



参考文献：

[1] 张斌，徐超，冯登国．流密码的设计与分析：回顾、现状与展望 [J]. 密码学报, 2016, 3 (6): 527-545. (Zhang Bin, Xuchao, Feng Dengguo. Design and analysis of stream ciphers: past, present and future directions [J]. Journal of Cryptologic Research, 2016, 3 (6): 527-545)

[2] Avaroğlu E, Koyuncu İ, Özer A B, *et al.* Hybrid pseudo-random number generator for cryptographic systems [J]. Nonlinear Dynamics, 2015, 82 (1-2): 58-61.

[3] Wang Yong, Liu Zhaolong, Ma Jianbin, *et al.* A pseudorandom number generator based on piecewise logistic map [J]. Nonlinear Dynamics, 2016, 83 (4): 2373-2391.

[4] Blum M, Micali S. How to generate cryptographically strong sequences of pseudo random bits [C]// Proc of Symposium on Foundations of Computer Science. 2008: 112-117.

[5] Bernstein D J, Hamburg M, Krasnova A, *et al.* Elligator: elliptic-curve points indistinguishable from uniform random strings [C]// Proc of ACM SIGSAC Conference on Computer & Communications Security. New York: ACM Press, 2013: 967-980.

[6] Alexi W, Chor B, Goldreich O, *et al.* RSA and Rabin functions: certain parts are as hard as the whole [J]. SIAM Journal on Computing, 1988, 17 (2): 194-209.

[7] 刘文瑞．抗量子计算攻击密码体制发展分析 [J]. 通信技术, 2017, 50 (5): 1054-1059. (Liu Wenrui. Quantum Computing Attack Password System Development Analysis [J]. Communications Technology, 2017, 50 (5): 1054-1059.)

[8] Impagliazzo R, Naor M. Efficient cryptographic schemes provably as secure as subset sum [C]// Proc of Symposium on Foundations of Computer Science. 1989: 236-241.

[9] Fischer J B, Stern J. An efficient pseudo-random generator provably as secure as syndrome decoding [C]// Proc of International Conference on Theory and Application of Cryptographic Techniques. Springer-Verlag, 1996: 245-255.

[10] Gaborit P, Lauradoux C, Sendrier N. SYND: a fast code-based stream cipher with a security reduction [C]// Proc of IEEE International Symposium on Information Theory. IEEE Xplore, 2007: 186-190.

[11] Meziani M, Cayrel P L, Alaoui S M E Y. 2SC: an efficient code-based stream cipher [C]// Proc of International Conference on Information Security and Assurance. Berlin: Springer, 2011: 111-122.

[12] Gaborit P, Hauteville A, Tillich J P. RankSynd a PRNG based on rank metric [M]// Post-Quantum Cryptography. Springer International Publishing, 2016.

[13] Augot D, Finiasz M, Sendrier N. A Family of Fast Syndrome Based Cryptographic Hash Functions [C]// Proc of International Conference on Cryptology in Malaysia. Berlin: Springer, 2005: 64-83.

[14] Berlekamp E R, Mceliece R J, Van Tilborg H C A. On the inherent intractability of certain coding problems (Corresp.) [J]. IEEE Trans. inf. theory, 1978, 24 (3): 384-386.

[15] 任方，郑东．一种基于编码的数字签名算法的改进 [J]. 西安邮电

[16] 大学学报, 2015, 20 (5): 38-43. (Ren Fang, Zheng Dong. An improved code based digital signature algorithm [J]. Journal of XI’AN University of Post and Telecommunications, 2015, 20 (5): 38-43.)

[17] Bertoni G, Daemen J, Peeters M, *et al.* Sponge functions [J]. Ecrypt Hash Workshop, 2007.

[18] Overbeck R, Sendrier N. Code-based cryptography [M]// Post-Quantum Cryptography. Springer Berlin Heidelberg, 2009: 95-145.

[19] Shenghui Su, Tao Xie, Shuwang Lü. A provably secure non-iterative hash function resisting birthday attack [J] , Theoretical Computer Science, 2016, 654 (22): 128-142

[20] Finiasz M, Sendrier N. Security Bounds for the Design of Code-Based Cryptosystems [C]// Proc of International Conference on Theory and Application of Cryptology and Information Security. Proceedings. 2009: 88-105.

[21] Hellman M. A cryptanalytic time-memory trade-off [J]. IEEE Trans on Infomation Theory, 1980, 26 (4): 401-406.

[22] Rukhin A, Soto J, Nechvatal J, *et al.* A statistical test suite for random and pseudorandom number generators for cryptographic applications [J]. Applied Physics Letters, 2015, 22 (7): 1645-179.

[23] 韩蕊，张雪锋．基于高维猫映射的伪随机序列生成方法 [J]. 计算机工程与应用, 2016, 52 (10): 91-99. (Han Rui, Zhang Xuefeng. Pseudo-random sequence generating method based on high dimensional cat map [J]. Computer Engineering and Applications, 2016, 52 (10): 91-99.)